DAIS zkouška

2018 - současnost

Cílem tohoto dokumentu je shromáždit otázky od nejmenovaného učitele a odpovědi, které uznal k otázce. Jak již někteří ví, je vy víte kdo dost náročný. Vaše odpovědi musejí odpovídat k otázce. Za přidání něčeho, co on považuje za něco, na co se neptal popřípadě drobná chyba se rovná nedosáhnutí minima. Za tyto roky se nashromáždily různé výcucy, materiály k tomuto předmětu. Přesto bylo dost lidí, co tímto předmětem z 2. ročníku bojují.   
  
Otázka zní: když máme všechny materiály a vy víte kdo tvrdí, že otázky, které podává na zkouškách jsou z přednášky, jsou stále někteří, kteří neprošli?

Odpověď

Učitel

1. Někteří jsou názoru, že vyučující používá někdy RNG (náhodný generátor čísel, házení kostkami), podle toho dostanete body
2. Mnozí jsou tohoto názoru, že vyučující je precizní a každé slovo nebo věta týkající se otázce může ohrozit a rozhodnout o minimu
3. Je velice malá pravděpodobnost, že usmlouváte body navíc. Některým se to povedlo (osobně jsem slyšel, že student měl 29b → napsal email učitelovi, zda by se na tu jeho zkoušku nepodíval a nenašel nějaké body navíc → nakonec získal potřebný počet bodů), některým ne (viz můj případ (autor tohoto dokumentu) jsem měl 28b (pozn. je možné že jsem měl před konzultací o 3b méně - nepamatuji se) ze zkoušky → domluvil jsem se u zkoušejicího konzultaci a společně nahlédli na můj test, přes mé snažení jsem nedosáhl minima)
4. Občas se zdá, že si hraje se studenty, mluví většinou klidně, nikdo nikdy netvrdil, že by po někom řval. Z pohledu vypadá velice vážně, občas jako by Vás chtěl roztrhat (samozřejmě se snaží na otázky odpovědět a provést diskuzi).
5. Veškeré argumenty jsou na jeho straně, nesnažte se přesvědčit o opaku. Buď uznejte chybu nebo se zeptat a tím projevit u něho svou snahu (neočekávejte tím body, spíše informace, které Vám mohou pomoci k zvládnutí této zkoušky).

Studenti studium

1. Pro 2. ročník je to celkem tento předmět terno a jedna z TOP 5 předmětů, na kterých studenti vyhořeli. Ve výsledku buď opakovali ročník nebo se dostali do 3. ročníku, ale ne skrze DAIS. No a někteří odešli z VŠB
2. Nepočítejte vždycky s tím, že na stránkách na které se odkazuje již zmíněný předmět má všechny přednášky → je třeba se nahlédnout do archívu a kliknout do předmětu.
3. Zjistilo se, že přednášky se některé aktualizují. Jinými slovy přednáška dva roky stará může obsahovat ménně i více než ta současná
4. Některé studijní materiály výpisky, výcucy a jiné sice vychází ze zdrojů knihy, přednášek. Ale nikdy neobsahovali vyložené otázky a odpovědi, které uznal. Nehledě na to, že si dost lidí i myslí, že i když dostanou stejnou otázku a odpoví stejně = výsledky se mohou lišit z 9b na 7b nebo dokonce 0!
5. Studenti mezi sebou nespolupracují a jedou na vlastní pěst. Tato strategie je jen Vaše věc, ale za časů studenti mezi sebou komunikovali a společně sbírali odpovědi, výsledky do každých předmětů. To se poněkud vytratilo, aktivita ve skupinách na fb se snížila. Víceménně dotazy k danému předmětu nejsou zodpovězeny, hledání studijních materiálů (pokud někdo nejde na všbsklad, kde je téměř vše)

Řešením by měla být taková to dokumentace + přednášky, popřípadě zápisky, pokud chodíte na přednášky.

**AKTUALIZACE: cituji zkoušejicího, na zkoušku dává jen ty otázky, které naleznete na přednáškách a o kterých mluvil. Pokud se nebavilo např. o bitmapovému indexu → nemůže to po nás chtít. (A na další věci otázek z minulých let tudíž nemusejí být)   
Na otázky, které dostáváte na zkoušce, a odpovědi, které chce zkoušející, jsou v přednáškách.**

Verdiktem naučit se vše co je na těch přednáškách, nejlépe ty věty, které tam jsou dát do zkoušky. Pokud bude otázka ohledně projektu ti co měli v minulosti nebo dělají hned. Buď si sežente od kolegy nebo vymyslete třeba svojí. Minispecifikace apod.

Někdy se dostalo příklad B-strom, kde jste měli vložit, kreslit → pohledat třeba časem naleznete v tomto dokumentu.

Tento dokument čerpá z několik již existujících materiálů. Výhodou tohoto dokumentu je, že skupina z nás šla za daným učitelem tohoto předmětu a vyfotili si svojí zkoušku. Proto apeluji do budoucna. Pokud jste nedali zkoušku a zjistíte, že ať už otázka není v této dokumentaci nebo odpovědi k dané otázce byla chybná (ztratili jste buď všechny nebo pár bodů). Opravte tento dokument popřípadě doplňte přesné otázky. Pokud v dokumentu se nacházejí odpovědi k otázce odkazujte se.

Tabulky tato aplikace má, tak ji použijte. Rovnice a další funkce.

Nevkládejte obrázky! Obrázek, který bychom uvítali v dokumentaci jsou: B-strom, Binární vyhlédávácí strom, kde se vkládají hodnoty zkrátka to, co je těžší.

Obrázky zvyšují velikost tohoto dokumentu! (Zkus jsme myslet stejně tak, jako chce učitel z DAIS, abychom přemýšleli o tomto předmětu) a nepoužili dostupné věci.

Berte na vědomí, že někteří studenti si to budou tisknout, aby měli něco po ruce. Sice jsme generace, která používá IT techniku, přesto bych doporučil:

A) psát si to dělat si poznámky, tím se to učit.

B) Vytisknout a číst

Nezapomeňte aktualizovat osnovu.

Nezapomeňte každý detail k dané otázce doplnit tu nejlépe možnou odpověď.

**DOPORUČUJI** - jak je to v případě [2018 Skupina A](#_heading=h.1t3h5sf) udělat vždycky to samé, rok skupina (možná i datum, pokud to bude nutné). Vložit otázky (najít možné odpovědi, buď v tomto dokumentu a odkazovat, nebo napsat takovou, kterou cvičící uznal). A v opačném případě, tam kde jste to našli vytvořit odkaz, aby se věděli, kdy se takové otázky dostali. Pomůže vám to třeba najít i studenta, který absolvoval předmět, ten Vám může poslat další materiály.

**Velice doporučuji si najít kolegy, sednout probrat získat informace, jeden ať zapisuje/aktualizuje, druhý hledá, třetí ověřuje, čtvrtý (pokud máte) komunikuje přes chat s dalšími.**

Další možnosti: můžete psát do chatu, můžete sem vložit i jen komentář, popřípadě je tu možnost editor pro minispecifikace nebo psaní kódu (dokument může být zdokonalován)

Upozornění: Za žádnou cenu nemazat obsah dokumentu nebo dokonce odstranění.

Za A) je tu vedena historie, kdo co konkrétně udělal, změnil

Za B) bude existovat kopie pro všechny možné případy

Děkuji za pochopení

**Osnova**

[**2018 zadání A**](#_heading=h.1t3h5sf) **5**

[K čemu v databázových systémech slouží log soubor? Popište proč je efektivnější zápis do logu než zápis do databáze.](#_heading=h.4d34og8) 5

[Popište problém souběhu zvaný výskyt fantomů, uveďte i příklad.](#_heading=h.2s8eyo1) 5

[Uveďte asymptotické časové složitosti pro tabulku typu halda a index typu B-strom pro operace vložení záznamu, smazání záznamu a bodový a rozsahový dotaz.](#_heading=h.17dp8vu) 6

[Co je to složený index? Kdy jej používáme? Popište výhody i nevýhody, uveďte příklad.](#_heading=h.3rdcrjn) 6

[Popište objektově-relační datový model, uveďte 4-5 nových rysů oproti relačnímu datovém modelu, uveďte příklady použití jednotlivých rysů.](#_heading=h.26in1rg) 7

[Napište minispecifikaci zvolené netriviální funkce z vašeho semestrálního projektu obsahující alespoň tři databázové operace (nepište: převod částky z účtu, kaskádové mazání atd.). Popište případně vstupní parametry i výstup funkce.](#_heading=h.lnxbz9) 8

[**Tato čára dělí horní část na otázky, které jsou přímo získány studenty, text otázek existuje. Některé zapsané odpovědi byli skutečně hodnoceny plnými počty, na některé otázky byly zatím vloženy, co jsme našli, jsou tam odkazy, které odkazují na dolní část, kde je možné témata a odpovědi + variace otázek, které můžete dostat a zatím neznáme buď přesnou formulaci, ať už otázek nebo odpovědí**](#_heading=h.sqyw64) **9**

[**Možné otázky na zkoušce a studijní materiály**](#_heading=h.1ksv4uv) **10**

[Statické/Dynamické PL/SQL, výhody a nevýhody (Napište rozdíl mezi statickým a dynamickým PL/SQL, výhody nevýhody a příklady obou), (viz. ---- Odpověď Rozdíl mezi statickým a dynamickým SQL. )](#_heading=h.44sinio) 10

[---- Odpověď](#_heading=h.2jxsxqh) 10

[PL/SQL vs T-SQL](#_heading=h.z337ya) 10

[Co je to kurzor? K čemu se využívá? Výhody a nevýhody. Uveďte příklad](#_heading=h.3j2qqm3) 11

[Popište odloženou aktualizaci, zda se jedná o UNDO nebo REDO.](#_heading=h.1y810tw) 11

[**Tabulka Izolace a fenomény (tabulka)**](#_heading=h.4i7ojhp) **11**

[Fenomén DIRTY READ (Špinavé čtení)](#_heading=h.2xcytpi) 13

[Fenomén NON-REPEATABLE READ (Neopakovatelné čtení)](#_heading=h.1ci93xb) 13

[Fenomén PHANTOM​ (Fantom)](#_heading=h.3whwml4) 13

[Rozdíl mezi Repeatable read a serializable](#_heading=h.2bn6wsx) 14

[Co nemůže nastat u REPEATABLE READ oproti READ COMMITED?](#_heading=h.qsh70q) 14

[NON-REPEATABLE READ (Neopakovatelné čtení)](#_heading=h.3as4poj) 14

[Co může nastat u read uncommited a nemůže nastat u read commited - naznačte na plánu dvou transakci.](#_heading=h.1pxezwc) 14

[Popište plán dotazu SELECT \* FROM tabulka WHERE login = abcde, kde login je index a tabulka je heap table.](#_heading=h.49x2ik5) 15

[Nakreslete binární vyhledávací strom po vložení čísel 20, 14, 5, 10, 8, 20, 3.](#_heading=h.2p2csry) 15

[Popište, co je SQL injection a jak mu zabránit.](#_heading=h.vx1227) 16

[SQL injection je spuštění SQL dotazu obsahujícího neošetřený kód od klienta, který může vést ke kompromitaci databáze.](#_heading=h.3fwokq0) 16

[Jakým způsobem lze u ORM minimalizovat počet přístupů k databázi, případně omezit zatížení sítě?](#_heading=h.4f1mdlm) 17

[Co je to ACID?](#_heading=h.2u6wntf) 17

[Halda, vkládání, vyhledávání a jejich složitost. (otázka o složitosti viz. Otázka skupiny A v roce 2018)](#_heading=h.19c6y18) 17

[Napište 4-5 vlastností objektově-relačního modelu (ve srovnání s relačním modelem). viz. Otázka skupiny A v roce 2018](#_heading=h.3tbugp1) 18

[Co je to doménový objekt a DAO v ORM? (Uvedtě příklad)](#_heading=h.28h4qwu) 18

[Aplikační objekty v ORM](#_heading=h.nmf14n) 18

[Seriový plán, serializovatelný plán](#_heading=h.37m2jsg) 18

[Popište plán dotazu SELECT \* FROM tabulka WHERE login = abcde, kde login je index a tabulka je heap table.](#_heading=h.1mrcu09) 19

[Popsat plán vykonání dotazu "SELECT \* FROM student WHERE login=@login", kdy login je indexem.](#_heading=h.46r0co2) 19

[Kandidát na index ( výhody X nevýhody )](#_heading=h.2lwamvv) 19

[**Literatura**](#_heading=h.111kx3o) **20**

[Kniha Databázové systémů - zde naleznete vše o databázových systémech. Zahrnuje předměty jako UDBS, DAIS, VIS a možná další nezmíněné předměty.](#_heading=h.206ipza) 20

[Přednášky z předmětu - Pozor někdy v daném roce chybí i přednášky → nespoléhat to co je dáno v daném roce, to bude na zkoušce a to co nebylo nebude. Dívat se do archívu.](#_heading=h.4k668n3) 20

[Jedna z variant, jak získat potřebné minimum](#_heading=h.2zbgiuw) 20

[Knihovna Dais Challengers - díky tomuto odkazu se dostanou studenti na disk, kde jsou uloženy všechny materiály, včetně tohoto dokumentu, který by měl mít v budoucnu hlavu a patu, všechny možné variace otázek a odpovědí.  
Jsou tam otázky, výcucy, výpisky z let minulých.](#_heading=h.1egqt2p) 21

[V případě problému nebo nejasností kontaktovat](#_heading=h.3cqmetx) 22

[(Zde pro každý rok, bych prosil, aby se podepsal, jako zvolený správce dokumentu - abych mohl dohledat a v případě nejasností vyřešit problém)](#_heading=h.2dlolyb) 22

# 

# **2018 zadání A**

Příklad 1 [9b]

## **K čemu v databázových systémech slouží log soubor? Popište proč je efektivnější zápis do logu než zápis do databáze.**

Slouží během zotavení (po chybě média) databáze, která není v korektním stavu na korektní stav.

Log soubor - je efektivnější, jelikož využívá sekvenčního zápisu, zatímco zápis do databáze má náhodný přístup na disk.

Jedná se o soubor - zápis je sekvenční

Zápis do databáze - je náhodný, rozloženo po 8kb

---

Systém je pak schopen provést zotavení databáze - UNDO, REDO operace

Do logu jsou v případě odložené aktualizace zapsány nové hodnoty, což umožní systému při zotavení realizovat operaci REDO.

V případě okamžité aktualizace ukládáme do logu původní hodnoty, což umožní systému provést při zotavení operaci UNDO

Příklad 2 [9b]

## **Popište problém souběhu zvaný výskyt fantomů, uveďte i příklad.**

Jedná se o problém kdy dvě transakce běžící ve stejné chvíli se ovlivňují.

Transakce A přečte počet řádků

Transakce B provede INSERT nebo DELETE

Později Transakce A přečte jiný počet řádků než dříve

t1 select count(\*) from user where year = 2000 (tři vysledky)

t2 insert into user (id, year) values (16, 2000)

t3 select count(\*) from user where year = 2009 (čtyři vysledky)

Příklad 3 [10b]

## **Uveďte asymptotické časové složitosti pro tabulku typu halda a index typu B-strom pro operace vložení záznamu, smazání záznamu a bodový a rozsahový dotaz.**

Tabulka typu halda má složitost vkládání O(1) pro hledání O(N) pro mazání O(N), bodový rozsah O(N)

Tabulka typu index má pro všechny operace složitost O(log n)

Příklad č. 4 [9b]

## **Co je to složený index? Kdy jej používáme? Popište výhody i nevýhody, uveďte příklad.**

Složený index (composite index) – index, kde je klíčem více než jeden atribut.

Výhody:

o 20% rychlejší vyhledávání nad indexem než u jednoduchého klíče.

Nevýhody:

Problém: Při indexu login,jmeno a dotazu na jméno, bude docházet k sekvenčnímu pr ůchodu!

Pokud se chceme dotazovat na hodnoty dvou a více atributů v jednom dotazu, můžeme vytvořit tzv. složený index.

Jednoduchý Index - je klíč jedním atributem

CREATE INDEX Producer\_name ON Producer(name);

Velikost indexu je 384 bloků (60% velikosti tabulky - 640 bloků)

Dotaz: SELECT \* FROM Producer WHERE name=’prod7452’;

Velikost výsledku: 1

consistent gets: 3 (logické přístupy)

Nevýhodou je, že zabírá příliš moc místa

čím více indexů tím větší režie

efektivita SELECT

snižování efektivity pro INSERT, DELETE, UPDATE

indexy brát s rozumem

Create index Producer\_name\_addr ON Producer(name, address);

Velikost indexu je 512 bloků (80% velikosti tabulky -- 640 bloků, velikost index Producer(name) je 384 bloků)

Dotaz: SELECT \* FROM Producer WHERE address = ’address56000’;

consistent gets: 574 (logické přístupy) !

Vykonávání tohoto dotazu se sekvenční vyhledávání.

Příklad č. 5 [9b]

## **Popište objektově-relační datový model, uveďte 4-5 nových rysů oproti relačnímu datovém modelu, uveďte příklady použití jednotlivých rysů.**

Nákladná migrace relačních dat, proto vznikly objektově-relační rysy v relačních databázích (SQL99). ⇒ Mluvíme o tzv. objektově-relačním datovém modelu (ORDM)

Objektové typy a jejich metody jsou uloženy spolu s daty v databázi, programátor tedy nemusí vytvářet podobné struktury v každé aplikaci.

Programátor může přistupovat k množině objektů jako by se jednalo o jeden objekt.

Objekty mohou jednoduše reprezentovat vazby kdy jedna entita se skládá z jiných entit (bez nutnosti použít vazeb).

Metody jsou spouštěny na serveru – nedochází k neefektivnímu přenosu dat po síti.

*(Zřejmě chtěl tohle, výše je podrobnější popis pozn. ne všechny)*

Vlastní (složené) strukturované datové typy.

Funkce, procedury, triggery a další kód na straně serveru.

Kolekce dat (vnořené tabulky, pole).

Dědičnost datových typů.

Datové typy ukazatelů na data.

Můžeme datovým typům přiřazovat chování (metody).

viz. [odkaz](#_heading=h.3tbugp1)

Přiklad č. 6 [9b]

## **Napište minispecifikaci zvolené netriviální funkce z vašeho semestrálního projektu obsahující alespoň tři databázové operace (nepište: převod částky z účtu, kaskádové mazání atd.). Popište případně vstupní parametry i výstup funkce.**

Vstupy

Vystupy

Popisek funkce

select

* popsat ho

insert

* popsat

dalši select

* popsat

Tato čára dělí horní část na otázky, které jsou přímo získány studenty, text otázek existuje. Některé zapsané odpovědi byli skutečně hodnoceny plnými počty, na některé otázky byly zatím vloženy, co jsme našli, jsou tam odkazy, které odkazují na dolní část, kde je možné témata a odpovědi + variace otázek, které můžete dostat a zatím neznáme buď přesnou formulaci, ať už otázek nebo odpovědí

# 

# Možné otázky na zkoušce a studijní materiály

## Statické/Dynamické PL/SQL, výhody a nevýhody (Napište rozdíl mezi statickým a dynamickým PL/SQL, výhody nevýhody a příklady obou), (viz. [---- Odpověď](#_heading=h.2jxsxqh) Rozdíl mezi statickým a dynamickým SQL. )

Statické - používají se statické příkazy:

DML - SELECT, UPDATE, DELETE, MERGE, INSERT, UPDATE

DCL - SET TRANSACTION, COMMIT, ROLLBACK, SAVEPOINT, LOCK TABLE

**Nevýhoda**

Nemožné používat DDL příkazy CREATE, ALTER, DROP, RENAME, TRUNCATE, COMMENT (a příkazy které v době překladu nejsou známy)

Dynamické - používají se tedy, jak nejsou jasné, jaké bude mít příkaz syntaxi - syntatické chyby. Používáme jen tehdy, jak není možné použít statické příkazy.

**Nevýhoda**

složité odhalování správnosti schématiky - možné chyby

možnost SQL Injection

Spuštění prostřednictvím: ExecuteImmediate

### ---- Odpověď

Statické SQL je kompilováno a používá implicitně parametrizované dotazy.

Používá se například v procedurálních rozšířeních DBŘS jako PL/SQL a T-SQL nebo v SQL rozšířeních jiných jazyků (vložené SQL v C).

Dynamické SQL se oproti tomu zasílá na DBŘS až v době jeho běhu, nelze tedy předvídat, jak dotazy budou vypadat a výkonnost je nižší, dotazy lze ale tvořit libovolně (jako řetězec), což sice může přinášet bezpečnostní problémy, ale dovoluje tvořit libovolné dotazy (JDD - definice dat).

## PL/SQL vs T-SQL

T-SQL neposkytuje operátory %TYPE, %ROWTYPE (namísto %ROWTYPE můžeme použít dočasnou pamět’ovou tabulku (temporary table), kterou označujeme prefixem #)

T-SQL nepodporuje CREATE OR REPLACE PROCEDURE, což nás nutí k zápisu:

/\*CREATE\*/ ALTER PROCEDURE

(od verze 2016 můžeme používat CREATE OR ALTER PROCEDURE)

T-SQL omezuje konstrukce, které můžeme využívat u funkcí

V T-SQL musíme pro práci s kurzory používat OPEN, FETCH, CLOSE, DEALLOCATE

T-SQL nás nutí k dvojitému FETCH u kurzorů

V T-SQL musíme definovat u parametrů procedur a funkcí délku datového typu (pokud se u datového typu udává)

## Co je to kurzor? K čemu se využívá? Výhody a nevýhody. Uveďte příklad

Kurzory jsou pomocné proměnné nějakého SQL příkazu

Umožní uchovávat jeden záznam, co se týká selektu.

Existují dva typy

implicitní - vytváří se automaticky po provedení příkazu INSERT, UPDATE, DELETE

explicitní - definuje se v definiční části procedury podobně jako proměnná, takový kurzor je často spojen s příkazem SELECT, který vrací více než jeden řádek

Pomocí kurzoru můžeme postupně procházet jednotlivé záznamy výsledku SELECT příkazu a pracovat s jednotlivými hodnotami

CURSOR jmeno\_kursoru IS prikaz\_select;

Kde prikaz\_select vrací množinu záznamů

## Popište odloženou aktualizaci, zda se jedná o UNDO nebo REDO.

Data se zapisují do paměťového bufferu, při kontrolním bodu jsou zapsána do logu a poté na disk.

Není třeba provádět UNDO, protože nepotvrzená data nejsou v databázi.

Pokud jsou data v logu a ne v databázi, provede se REDO.

# Tabulka Izolace a fenomény (tabulka)

**Úroveň izolace**

Izolovanost spotřebovává nemalé prostředky SŘBD a snižuje propustnost.

SŘBD nabízí možnost regulace míry izolovaností vůči propustnosti. Existují 4 úrovně:

|  |  |  |  |
| --- | --- | --- | --- |
| stupeň izolace/fenomén | **DIRTY READ** | **NON-REPEATABLE READ** | **PHANTOM** |
| **READ UNCOMMITTED (RU)** | může nastat | může nastat | může nastat |
| **READ COMMITTED (RC)** | nemůže nastat | může nastat | může nastat |
| **REPEATABLE READ (RR)** | nemůže nastat | nemůže nastat | může nastat |
| **SERIALIZABLE (SR)** | nemůže nastat | nemůže nastat | nemůže nastat |

**READ UNCOMMITTED (RU)**

Dovoluje číst nepotvrzení změny ostatních transakcí, nebezpečné (nejrychlejší)

Nejnižší úroveň (stupeň izolovanosti) – Nejvyšší propustnost a žádná izolovanost.

Je pouze READ ONLY!

**Problémy izolovanosti - výjimky souběhu**

Může nastat: [Dirty Read](#_heading=h.2xcytpi) (špatné čtení), [Non-Repeatable Read](#_heading=h.1ci93xb) (neopakovatelné čtení), [Phatom](#_heading=h.3whwml4) (fantom)

**READ COMMITTED (RC)**

Dovoluje číst potvrzené změny ostatních transakcí (zámky se mohou uvolnit před koncem transakce)

Zámky S se mohou uvolnit ještě dříve než dojde k ukončení transakce.

Zámky X se uvolní až po ukončení transakce.

**Problémy izolovanosti - výjimky souběhu**

Může nastat: [Non-Repeatable Read](#_heading=h.3as4poj) (neopakovatelné čtení), [Phantom](#_heading=h.3whwml4) (fantom)

**REPEATABLE READ (RR)**

Úroveň serializace, která nedovoluje špinavá čtení ani problém nekonzistenční analýzy. Stále se ale v ní může objevovat problém fantomů, tj. nově přidaných dat do databáze, který může ovlivnit výsledky dotazů.

Zámky S i X se uvolní až po ukončení transakce.

[Nebo]

zajišťuje opakovatelnost čtení, ale můžou se zde vyskytovat fantomové (nově přidané záznamy)

**Problémy izolovanosti - výjimky souběhu**

Může nastat: [Phantom](#_heading=h.3whwml4) (fantom)

**SERIALIZABLE (SR)**

Zajišťuje kompletní izolovanost transakce od ostatních transakcí, jako jediná úroveň plně zajišťuje vlastnost ACID (nejpomalejší)

Nejvyšší úroveň – Nejnižší propustnost a vysoká/maximální izolovanost.

Zámky S i X se uvolní až po ukončení transakce.

Díky této izolovanosti se nemůže nastat [Dirty Read](#_heading=h.2xcytpi), [Non-Repeatable Read](#_heading=h.1ci93xb) a [Phantom](#_heading=h.3whwml4)

**Problémy izolovanosti - výjimky souběhu**

Maximální stupeň izolovanosti, ale nižší propustnost. / Žádné – proběhne korektní izolace.

### Fenomén DIRTY READ (Špinavé čtení)

Klient A provede změnu dat a prozatím neukončí transakci.

Klient B přečte tato změněná data.

Poté klient A odvolá svou transakci.

Klient B tedy přečetl data, která nikdy nebyla potvrzena.

[nebo]

Transakce A dostane nepotvrzená data, na kterých jiná transakce B zatím neprovedla

COMMIT nebo ROLLBACK.

### Fenomén NON-REPEATABLE READ (Neopakovatelné čtení)

Klient A přečte data a prozatím neukončí transakci.

Klient B změní nebo zruší tato data a ukončí svou transakci.

Klient A ve své transakci znovu čte stejná data a nenajde je.

[nebo]

Transakce při opakovatelném spouštění stejného dotazu (SELECT), dostane jiný záznam.

Před dalším spuštěním, jiná transakce provede UPDATE.

### Fenomén PHANTOM​ (Fantom)

(viz. [Popište problém souběhu zvaný výskyt fantomů, uveďte i příklad.](#_heading=h.2s8eyo1))

Klient A položí dotaz, přečte odpověď na něj a prozatím neukončí transakci.

Klient B vloží do databáze další řádky vyhovující podmínkám v dotazu klienta A a ukončí svou transakci.

Klient A ve své transakci znovu položí stejný dotaz a obdrží jinou odpověď.

[nebo]

Transakce při opakovatelném spouštění stejného dotazu (SELECT), dostane jiný počet

záznamů.

Před dalším spuštěním, jiná transakce provede INSERT.

## Rozdíl mezi Repeatable read a serializable

U úrovně izolace Serializable (SR) jsou všechny řádky uzamčeny po dobu trvání transakce.

U RR je možné vkládání nových řádku do datového souboru.

Repeatable read RR:

K uvolnění zámku S a X dochází až po ukončení transakce

Problémy: výskyt fantomů = pokud provedeme stejný dotaz ( select) v transakci vícekrát získáme jiný počet záznamů

Serializable (SR):

K uvolnění zámku S a X dochází až po ukončení transakce

Nejvyšší stupeň izolace

Snižuje propustnost databaze

Nemá problémy

## Co nemůže nastat u REPEATABLE READ oproti READ COMMITED?

## [NON-REPEATABLE READ (Neopakovatelné čtení)](#_heading=h.1ci93xb)

## Co může nastat u read uncommited a nemůže nastat u read commited - naznačte na plánu dvou transakci.

[**DIRTY READ (Špinavé čtení)**](#_heading=h.2xcytpi)

|  |  |  |
| --- | --- | --- |
| Transakce A | čas | Transakce B |
| - | t2 | UPDATE student SET rocnik = 1 WHERE id=’jir0081’ |
| SELECT \* FROM student WHERE id=’jir0081’ | t3 | - |
| COMMIT | t4 | - |
|  | t5 | COMMIT/ROLLBACK |

viz. [Tabulka Izolace a fenomény (tabulka)](#_heading=h.1rvwp1q)

## Popište plán dotazu SELECT \* FROM tabulka WHERE login = abcde, kde login je index a tabulka je heap table.

V indexu se nalezne prvek s loginem abcde (průchod B+ stromem, rychlé, ale náhodný přístup k datům na disku/v bufferu) - UNIQUE SCAN

Podle ROWID z indexu se načte záznam z tabulky a vyberou se všechny atributy - TABLE ACCESS BY ROWID / RID Lookup

## Nakreslete binární vyhledávací strom po vložení čísel 20, 14, 5, 10, 8, 20, 3.

## 

## 

## 

## 

## 

## 

## 

## 

## Popište, co je SQL injection a jak mu zabránit.

## SQL injection je spuštění SQL dotazu obsahujícího neošetřený kód od klienta, který může vést ke kompromitaci databáze.

Útočník může vkládat SQL dotaz do formuláře nebo zadat do URL

Příklad:

Dotaz = "SELECT \* FROM [user] WHERE login = '" + login + "' AND pass = '" + pass + "'" Execute(Dotaz)

pokud uživatel (např. ve formuláři) vyplní do loginu např. řetězec

asd' OR 1=1; --

tak se vrátí první uživatel z databáze (a klient se na něho přihlásí), i když nezná jeho přihlašovací údaje.

**Pokud by se ptal jak tomu zabránit (nejsem si jistý, někdy se může zeptat jen co je SQL injection)**

1. Sanizace vstupních řetězců - nebezpečné, pokud to programujeme sami, téměř jistě na něco zapomeneme, ale i pokud používáme knihovny, může se stát, že velmi speciálně zkonstruovaný vstup (např. v nějaké čínském znakové sadě) bude moct injektovat škodlivý kód
2. Parametrizované dotazy (implicitní, pokud se používá statické SQL) - správné řešení, které nejprve na server odešle dotaz s "placeholdery" pro vstupní data, a až poté samotná data.zajistí se tak, že server bere vstup uživatele opravdu jako data, a ne jako direktivy dotazu - zároveň se tak dá optimalizovat spouštění více stejných dotazů s různými daty, protože server už má předpočtený plán dotazu a pouze ho spouští znovu nad různými daty (pokud je ale dat opravdu hodně, mělo by se spíše použít bulk vkládání)
3. Omezit přístupová práva - skript my měl mít co nejméně přístupových práv, tak, aby mohl provádět jen to, co má, a nemohl například zasahovat do ostatních částí databáze

## 

## Jakým způsobem lze u ORM minimalizovat počet přístupů k databázi, případně omezit zatížení sítě?

Měli bychom využít znalosti konkrétní databáze a navrhnout ORM tak, abychom stahovali data s co nejmenším počtem dotazů (je např. lepší použít jeden SELECT dotaz, který vrátí milion výsledků, než 1000 SELECT dotazů, kde každý vrátí 1000 výsledků).

Místo více dotazů tedy můžeme použít spojování tabulek. Měli bychom taky stahovat pouze ta data, které potřebujeme.

Pokud navrhujeme funkci, která generuje velký počet mezivýsledků, měla by být naimplementovaná na straně databáze (jako uložená procedura/funkce), aby se mezivýsledky nemusely zbytečně přenášet přes síť.

## Co je to ACID?

Každá transakce musí splňovat vlastnost ACID:

atomičnost (angl. atomicity),

korektnost (angl. correctness),

izolovanost (angl. isolation)

a trvalost (angl. durability):

**A - Atomičnost** – transakce musí být atomická: jsou provedeny všechny operace transakce nebo žádná.

**C - Korektnost** – transakce převádí korektní stav databáze do jiného korektního stavu databáze, mezi začátkem a koncem transakce nemusí být databáze v korektním stavu.

**I - Izolovanost** – transakce jsou navzájem izolovány: změny provedené jednou transakcí jsou pro ostatní transakce viditelné až po provedení COMMIT.

**D - Trvalost** – jakmile je transakce potvrzena, změny v databázi se stávají trvalými i po případném pádu systému.

## Halda, vkládání, vyhledávání a jejich složitost. (otázka o složitosti viz. [Otázka skupiny A v roce 2018](#_heading=h.17dp8vu))

Tabulka typu halda je stránkované perzistentní pole,

vkládání proběhne tak, že se záznam vloží na první prázdné místo, složitost tedy má O(1).

Vyhledávání je relativně složitá operace se složitostí O(n), musí se sekvenčně projít všechny záznamy.

(Tohle popisuje haldu)

tabulka typu halda (angl. heap table) – jedna se o stránkované sekvenční pole (dále v textu budeme mluvit jen o tabulce),

(Tohle popisuje index)

index – nejčastěji stránkovány B-strom nebo nějaká jeho varianta (např. B +-strom).

## Napište 4-5 vlastností objektově-relačního modelu (ve srovnání s relačním modelem). viz. [Otázka skupiny A v roce 2018](#_heading=h.26in1rg)

Vlastní (složené) strukturované datové typy.

Funkce, procedury, triggery a další kód na straně serveru.

Kolekce dat (vnořené tabulky, pole).

Dědičnost datových typů.

Datové typy ukazatelů na data.

Můžeme datovým typům přiřazovat chování (metody).

## Co je to doménový objekt a DAO v ORM? (Uvedtě příklad)

Doménový objekt reprezentuje objekt z domény aplikace (např. Auto v IS autopůjčovny).

DAO (data access object) poskytuje rozhraní k práci s DTO (data transfer object), které slouží jako přepravky dat a jsou mapovány na konkrétní záznam v tabulce.

DAO představuje obvykle tabulku samotnou a její funkce (vyhledání konkrétního záznamu, smazání konkrétního záznamu, načtení všech záznamů), DTO představuje jeden záznam (jeho data).

## Aplikační objekty v ORM

DTO (data transfer object) - objekt přenášející data. Představuje jeden záznam (řádek) z tabulky.

## Seriový plán, serializovatelný plán

Sériový plán - je vykonání transakcí za sebou sériově.

[nebo jinak] Sériový plán​ - transakce jsou provedeny za sebou

Serializovatelný plán - je ekvivalentní s libovolným sériovým plánem (viz teorie nahoře)

[nebo delší varianta]

výsledek tohoto plánu odpovídá výsledku libovolného sériového plánu (jinak řečeno, výsledek po spuštění transakcí paralelně odpovídá výsledku, který bychom dostali, kdybychom je spustili všechny po sobě, v libovolném pořadí)

- přísné dvoufázové zamykání zaručuje, že plán bude vždy serializovatelný

## Popište plán dotazu SELECT \* FROM tabulka WHERE login = abcde, kde login je index a tabulka je heap table.

1) V indexu se nalezne prvek s loginem abcde (průchod B+ stromem, rychlé, ale náhodný přístup k datům na disku/v bufferu) - UNIQUE SCAN

2) Podle ROWID z indexu se načte záznam z tabulky a vyberou se všechny atributy - TABLE ACCESS BY ROWID / RID Lookup

## Popsat plán vykonání dotazu "SELECT \* FROM student WHERE login=@login", kdy login je indexem.

V indexu (obvykle B+ strom) se najde klíč login, pokud je unikátní, jedná se o UNIQUE SCAN.

Pokud ne, tak se najde nejnižší hodnota a poté se jde dále pomocí zřetězených odkazů v listech (RANGE SCAN).

Takto se najde rowid záznamu v tabulce typu halda a k tomu se přistoupí a načtou se všechna data

## Kandidát na index ( výhody X nevýhody )

Primární nebo cizí klíč nebo atribut často se vyskytující v klauzuli where.

Výhody: rychlejší prohledávání tabulek.

Nevýhody: každý index znamená zvýšení počtu operací při změnách v databázi

# Literatura

Odkazy online-knihy, přednášky, stránky, ze kterých jste se ještě učili. Vložte. Snažme se, aby tento dokument se stal pro další generace zbraní na absolvování předmětu DAIS.

### [Kniha Databázové systémů](http://dbedu.cs.vsb.cz/SubPages/OpenFile.aspx?file=book/dbcb.pdf) - zde naleznete vše o databázových systémech. Zahrnuje předměty jako UDBS, DAIS, VIS a možná další nezmíněné předměty.

### [Přednášky z předmětu](http://dbedu.cs.vsb.cz) - Pozor někdy v daném roce chybí i přednášky → nespoléhat to co je dáno v daném roce, to bude na zkoušce a to co nebylo nebude. Dívat se do archívu.

(Pozn. přednášky z roku 2017-2018 hodím do knihovny [Dais Challengers](#_heading=h.1egqt2p) do složky Přednášky 2017-2018, abyste měli všechny přednášky k tomu roku.

Výhoda máte mobily, můžete si kromě tohoto dokumentu stáhnout obsah přednášky a zkusit [jednu z možných variant](#_heading=h.2zbgiuw)

Pokud v následujících letech se objeví nová prezentace → stáhnout a nahrát, nebo se aktuální přednášky změní → nevymazat aktuální nahrané soubory, nahrát nový soubor a pojmenovat s rokem, kdy byla tato přednáška objevena)

## Jedna z variant, jak získat potřebné minimum

**Podle dostupného zdroje: je jedna z možných efektivních řešení naučit se nazpaměť jeho přednášky a psát stručné, ale zcela jasné odpovědi. Naleznete je v zde**

**Poznámka: V podstatě je to jako control cizí a control vlastní, akorát v reálném životě (sarkasmus)**

### [Knihovna Dais Challengers](https://drive.google.com/open?id=1AKCoFyzy78ttMwgyhhAm3pzU5vGgPzU_) - díky tomuto odkazu se dostanou studenti na disk, kde jsou uloženy všechny materiály, včetně tohoto dokumentu, který by měl mít v budoucnu hlavu a patu, všechny možné variace otázek a odpovědí. Jsou tam otázky, výcucy, výpisky z let minulých.

Jak asi víte na FB jsou každý rok vytvořeny skupiny co se týká naší fakulty. Stačí napsat správcům o možnost přidání se do dané skupiny. Vyhledat topic (většinou DAIS, DAIS - zkouška)

Existují další uložiště, kde jsou materiály jako například dropbox (momentálně jsem nenašel, ale v podstatě materiály, jsou v [Knihovna Dais Challengers](https://drive.google.com/open?id=1AKCoFyzy78ttMwgyhhAm3pzU5vGgPzU_) - kdyby jste z dropboxu našlo, co není a bylo to užitečné tak nahrajte, jinak se snažte tento dokument obnovovat, aktualizovat)

### [Učebnice Databázové systémy 2 - z Trnavy](http://pdf.truni.sk/e-ucebnice/databazove-systemy2/) - tato učebnice zahranuje některé věci týkající databázích (možné i z DAIS)

k nalezení ’Distribuovaný databázový systém’ : [odkaz](http://pdf.truni.sk/e-ucebnice/databazove-systemy2/data/839db3f0-6971-43d5-bbbd-c697fcc4cfd6.html?ownapi=1)

V případě problému nebo nejasností kontaktovat

### (Zde pro každý rok, bych prosil, aby se podepsal, jako zvolený správce dokumentu - abych mohl dohledat a v případě nejasností vyřešit problém)

**Hlavní správce dokumentu Vojtěch Jíra** [**Facebook**](https://www.facebook.com/Adalbert.Giorgio)